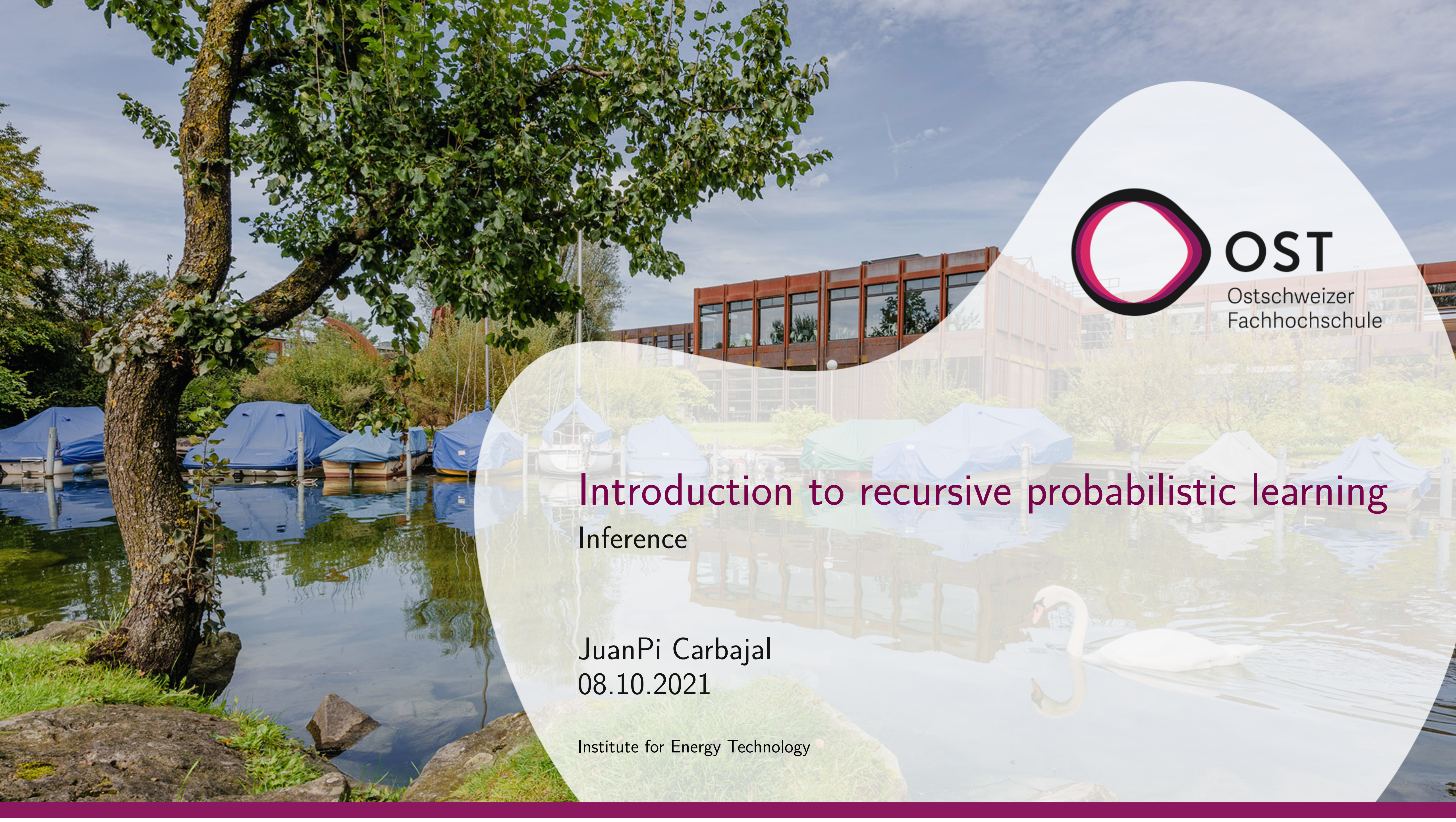


Introduction to recursive probabilistic learning

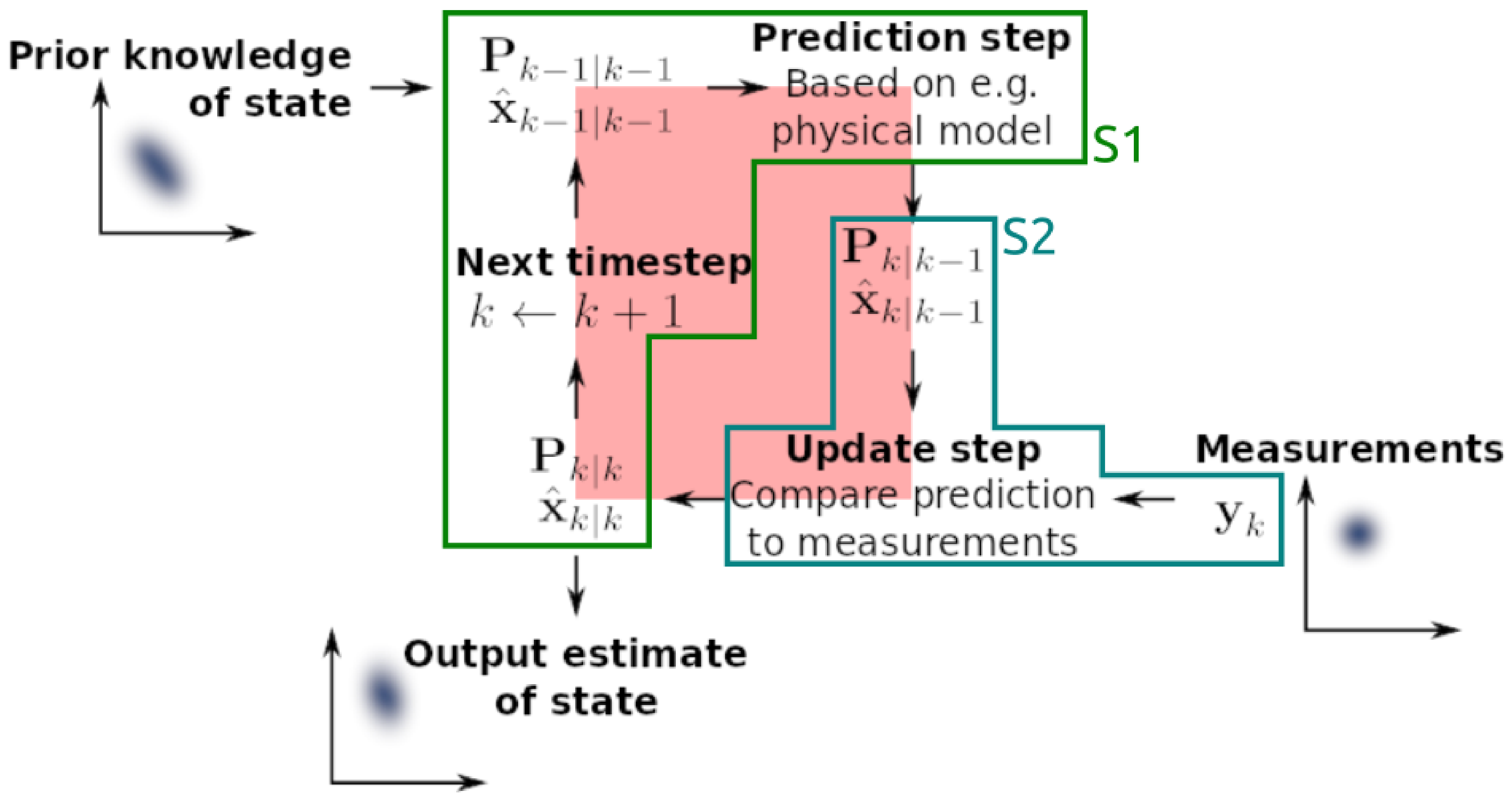
Inference

JuanPi Carbajal
08.10.2021

Institute for Energy Technology

Recap and Q&A

Sessions structures



Prediction step

The iteration represent a dynamical model of a process:

$$\begin{aligned} \boldsymbol{x}_k &= \mathbf{A}\boldsymbol{x}_{k-1} + \boldsymbol{\epsilon}_k \quad \boldsymbol{\epsilon}_k \sim \mathcal{N}(0, \mathbf{Q}_k) \\ \boldsymbol{y}_k &= \mathbf{H}\boldsymbol{x}_k + \boldsymbol{\rho}_k \quad \boldsymbol{\rho}_k \sim \mathcal{N}(0, \mathbf{R}_k) \end{aligned}$$

Mean and variance of forward predictions (prediction step)

$$\boldsymbol{m}_k = \mathbf{A}\boldsymbol{m}_{k-1} \qquad \mathbf{P}_k = \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^\top + \mathbf{Q}_{k-1}$$

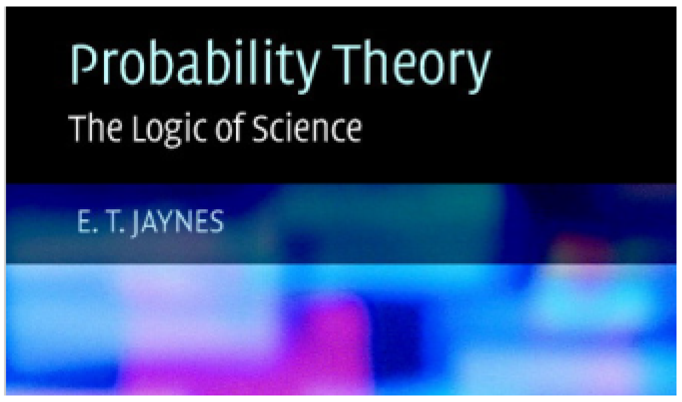
How do the predicted measurements look like?

(Conditional) Probability

Probability

The actual science of logic is conversant at present only with things either certain, impossible, or entirely doubtful, none of which (fortunately) we have to reason on. Therefore the true logic for this world is the calculus of Probabilities, which takes account of the magnitude of the probability which is, or ought to be, in a reasonable man's mind.

James Clerk Maxwell (1850)



Probability is a numerical between value assigned to a plausibility. Impossibility is assigned the probability 0 and certainty is assigned the probability 1.

We will see how Bayesian probability includes classical logic as a particular case and extends it to include reasoning on plausible events.

Propositional logic

Works with propositions (A, B, etc.) that can be true or false.

$$A \equiv \text{It rains at 10:00}$$
$$B \equiv \text{We see a blue sky}$$

Logic

- negation $\neg A$: true if A is false
- conjunction $A \wedge B$: true if A and B are both true
- disjunction $A \vee B$: true if any of (or both) A or B are true
- implication $A \implies B$: says that $A \wedge \neg B$ is false ($\neg A \vee B$ is true)

Boolean

- $\bar{A}$
- AB
- $A + B$
- $A \implies B$

State several propositions and determine their truth value. Does the value depend on the context?

Implication (logical)

$A \implies B$: says that $A\bar{B}$ is false ($\bar{A} + B$ is true).

Means **only that** $A = AB$ (the truth value of A is the same as the truth value of AB).

A	$\implies$	B
T	T	T
T	F	F
F	T	T
F	T	F

- All true propositions logically imply all other true propositions.
- It doesn't mean: B deducible from A. This depends on the background information.
- It doesn't mean: B is caused by A. Causal physical consequence can be effective only at a later time. The rain at 10:00 is not the physical cause of the clouds before 10:00. Implication doesn't follow the uncertain causal direction clouds$\rightarrow$rain, but rather the certain non-causal rain$\rightarrow$clouds.

Conditional probability

$$p(A|B)$$

The plausibility (probability) of a proposition A depends, in general, on whether we known/assume/believe some other proposition B is true.

Example:

$$\begin{aligned} A &\equiv \text{My english pronunciation is perfect} \\ B &\equiv \text{I grew up in Argentina} \\ C &\equiv \text{I grew up in England} \\ &\text{which is bigger? } p(A|B),\ p(A|C) \end{aligned}$$

All probabilities are conditional probabilities, because there is always some assumptions we made or knowledge we have that is relevant for the assignation.

Think of "The probability of a 6 in a fair die is $\frac{1}{6}$". What's the background information?

Product rule

To evaluate $p(AB|C)$ we can proceed as follows:

1. determine the probability of A true given information C. $\rightarrow p(A|C)$
2. based on that, determine the probability of B true. $\rightarrow p(B|AC)$

1. determine the probability of B true given information C. $\rightarrow p(B|C)$
2. based on that, determine the probability of A true. $\rightarrow p(A|BC)$

Both paths should give the same

$$p(AB|C) = p(A|BC) \overbrace{p(B|C)}^{\text{"prior" to knowing } A} = p(B|AC) \overbrace{p(A|C)}^{\text{"prior" to knowing } B}$$

$p(AB|C)$ is called the joint probability of A and B. The probability that both propositions are true.
There is nothing special in the "prior", it is just the probability we would assign "before" we knew the other piece of information. "before" here uses a temporal expression but it might have nothing to do with time.

Sum rule

$$p(A|C) + p(\bar{A}|C) = 1 \;\rightarrow\; \sum_i p(A_i|C) = 1 \;\rightarrow\; \int_{-\infty}^{\infty} p(A = a|C)\mathrm{d}a = 1$$

States that the probability distributes totally over the plausible propositions.

Particular case of (using $B = \bar{A}$):

$$p(A + B|C) = p(A|C) + p(B|C) - p(AB|C)$$

Syllogisms

source: *Luis Prade and Gan Khoon Lay at* `https://thenounproject.com`

A dark night, a policeman walks down an apparently deserted street. Suddenly burglar alarm, across the street a jewelry store with a broken window. A masked man crawls out of the broken window, carrying a bag which turns out to be full of expensive jewelry. The policeman doesn't hesitate at all in deciding that this man is dishonest.

Is guilt implied by evidence?

Is it a logical deduction? Alternative hypothesis?

Is there validity in the reasoning?

Syllogisms

Assume that $A \implies B$ is true.

Strong

- if A is true, then B is true

  $A \equiv$ rain at 10:00, $B \equiv$ clouds before 10:00
  $A \equiv$ man is burglar, $B \equiv$ robbed jewelry store

- if B is false, then A is false

  $\bar{B} \equiv$ no clouds before 10:00, $\bar{A} \equiv$ no rain at 10:00
  $\bar{B} \equiv$ didn't rob jewelry store, $\bar{A} \equiv$ man isn't burglar

Weak

- if B is true, then A is more plausible
  Evidence doesn't prove A, verification of its consequence gives more confidence in A

  $B \equiv$ clouds before 10:00, $A \equiv$ rain at 10:00
  $B \equiv$ robbed jewelry store, $A \equiv$ man is burglar

- if A is false, then B less plausible
  Evidence doesn't disprove B, one of the reasons eliminated, less confidence in B

  $\bar{A} \equiv$ no rain at 10:00, $B \equiv$ clouds before 10:00
  $\bar{A} \equiv$ man isn't burglar, $B \equiv$ robbed jewelry store

The implication does not imply causation or deduction. The examples are just for illustration of the syllogisms.

Syllogisms

Assume that $A \implies B$ is true.

$C \equiv A \implies B$

Strong

- if A is true, then B is true

$$p(B|AC) = \frac{p(AB|C)}{p(A|C)} = \frac{p(A|C)}{p(A|C)} = 1$$

- if B is false, then A is false

$$p(A|\bar{B}C) = \frac{\overbrace{p(A\bar{B}|C)}^{=0}}{p(\bar{B}|C)} = 0$$

Weak

- if B is true, then A is more plausible

$$p(A|BC) \geq p(A|C) \quad p(\bar{A}|BC) \leq p(\bar{A}|C)$$

$$\frac{\overbrace{p(B|AC)}^{=1}}{p(B|C)} \geq 1$$

- if A is false, then B less plausible

$$p(B|\bar{A}C) \leq p(B|C)$$

$$\frac{p(\bar{A}|BC)}{p(\bar{A}|C)} \leq 1$$

Weaker sillogism

Assume that "if A is true, then B is more plausible" is true.

if B is true, then A more plausible.

Assume that if "a man is robbing a jewelry store" it is more plausible that he will be "found in the suspicious situation". If he is actually found, then it is more plausible that he is robbing the jewelry.

We assume that

$$p(B|AZ) \geq p(B|Z)$$

then the product rule gives the result

$$p(A|BZ) = \frac{p(B|AZ)}{p(B|Z)} p(A|Z) \geq p(A|Z)$$

I used Z for the context information, to avoid confusion with the C used before.

Update step

Probabilistic prediction step

$$\begin{tikzcd}
{} \arrow[r] & X_{k-1} \arrow[r, "\mathbf{A}_k"] \arrow[d, "\mathbf{H}_{k-1}"'] & X_k \arrow[r] \arrow[d, "\mathbf{H}_k"'] & {} \\
& Y_{k-1} & Y_k &
\end{tikzcd}$$

Dynamic model (noisy): $p(x_k|x_{k-1})$

Given $p(x_{k-1}|y_{:k-1})$ it provides $p(x_k|x_{k-1}y_{:k-1})$.
The model uses only x_{k-1}, the previous state "blocks" information from previous measurements, hence $p(x_k|x_{k-1}y_{:k-1}) = p(x_k|x_{k-1})$

The previous states are not sharp values
but distributed, to condition on the measurement we "average" over all possible previous states, compatible with the measurements so far:

$$p(x_{\boldsymbol{k}}|y_{1:\boldsymbol{k-1}}) = \int p(x_k|x_{k-1})p(x_{k-1}|y_{1:k-1})\mathrm{d}x_{k-1}$$

Measurment model (noisy): $p(y_k|x_k)$

When the proposition $A \equiv$ the variable takes the value v we write

$$p(A|C) \doteq p(v|C)$$

I have omitted the background information in the slide, but it is always there. For example, the (parameters of the) distribution of the noise.
When y_k is given (measured), $p(y_k|x_k)$ is called the likelihood of the measured value (data).

Probabilisitc update step

Prediction

$$p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k-1}) = \int p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})p(\boldsymbol{x}_{k-1}|\boldsymbol{y}_{1:k-1})\mathrm{d}\boldsymbol{x}_{k-1}$$

Update

$$p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k}) \propto p(\boldsymbol{y}_k|\boldsymbol{x}_k)p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k-1})$$

- $p(\boldsymbol{x}_{k-1}|\boldsymbol{y}_{1:k-1})$ previous estimate.
- $p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k-1})$ what we believe $\boldsymbol{x}_k$ would be before we observed $\boldsymbol{y}_k$ (prediction).
- $p(\boldsymbol{y}_k|\boldsymbol{x}_k)$ the likelihood of the observed $\boldsymbol{y}_k$ given our prior belief (measurement).

Distribution of Jointly Gaussian variables

For Gaussian variables the computations can be written explicitly:

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{m},\mathbf{P}) = \frac{1}{\sqrt{(2\pi)^n \det \mathbf{P}}} \exp -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{m})^\top \mathbf{P}^{-1}(\boldsymbol{x}-\boldsymbol{m})$$

$$\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{y}\end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}\boldsymbol{a}\\ \boldsymbol{b}\end{bmatrix}, \begin{bmatrix}\mathbf{A} & \mathbf{C}\\ \mathbf{C}^\top & \mathbf{B}\end{bmatrix}\right)$$

$$\begin{aligned}\boldsymbol{x} &\sim \mathcal{N}(\boldsymbol{a},\mathbf{A})\\ \boldsymbol{y} &\sim \mathcal{N}(\boldsymbol{b},\mathbf{B})\\ \boldsymbol{x}|\boldsymbol{y} &\sim \mathcal{N}(\boldsymbol{a}+\mathbf{C}\mathbf{B}^{-1}(\boldsymbol{y}-\boldsymbol{b}), \mathbf{A}-\mathbf{C}\mathbf{B}^{-1}\mathbf{C}^\top)\\ \boldsymbol{y}|\boldsymbol{x} &\sim \mathcal{N}(\boldsymbol{b}+\mathbf{C}^\top\mathbf{A}^{-1}(\boldsymbol{x}-\boldsymbol{a}), \mathbf{B}-\mathbf{C}^\top\mathbf{A}^{-1}\mathbf{C})\end{aligned}$$

$$\begin{aligned}\boldsymbol{x} &\sim \mathcal{N}(\boldsymbol{m},\mathbf{P})\\ \boldsymbol{y}|\boldsymbol{x} &\sim \mathcal{N}(\mathbf{H}\boldsymbol{x}+\boldsymbol{u},\mathbf{R})\end{aligned}$$

$$\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{y}\end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}\boldsymbol{m}\\ \mathbf{H}\boldsymbol{x}+\boldsymbol{u}\end{bmatrix}, \begin{bmatrix}\mathbf{P} & \mathbf{P}\mathbf{H}^\top\\ \mathbf{H}\mathbf{P} & \mathbf{H}\mathbf{P}\mathbf{H}^\top+\mathbf{R}\end{bmatrix}\right)$$

$$\boldsymbol{y} \sim \mathcal{N}(\mathbf{H}\boldsymbol{x}+\boldsymbol{u}, \mathbf{H}\mathbf{P}\mathbf{H}^\top+\mathbf{R})$$

Do exercise 3.5 from Särkkä, S. (2013). Bayesian Filtering and Smoothing doi:10.1017/CBO9781139344203
Check the Schur complement of a matrix.

Prediction and Update step

Prediction

$$p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k-1}) = \int p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})p(\boldsymbol{x}_{k-1}|\boldsymbol{y}_{1:k-1})\mathrm{d}\boldsymbol{x}_{k-1}$$

$$\begin{aligned}\hat{\boldsymbol{m}}_k &= \mathbf{A}_k\boldsymbol{m}_{k-1}\\ \hat{\mathbf{P}}_k &= \mathbf{A}_k\mathbf{P}_{k-1}\mathbf{A}_k^\top + \mathbf{Q}_{k-1}\end{aligned}$$

Update

$$p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k}) \propto p(\boldsymbol{y}_k|\boldsymbol{x}_k)p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k-1})$$

$$\begin{aligned}\hat{\boldsymbol{y}}_k &= \mathbf{H}_k\hat{\boldsymbol{m}}_k\\ \mathbf{S}_k &= \mathbf{H}_k\hat{\mathbf{P}}_k\mathbf{H}_k^\top + \mathbf{R}_k\\ \mathbf{K}_k &= \hat{\mathbf{P}}_k\mathbf{H}_k^\top\mathbf{S}_k^{-1}\\ \boldsymbol{m}_k &= \hat{\boldsymbol{m}}_k + \mathbf{K}_k\left(\boldsymbol{y}_k - \hat{\boldsymbol{y}}_k\right)\\ \mathbf{P}_k &= \hat{\mathbf{P}}_k - \mathbf{K}_k\mathbf{S}_k\mathbf{K}_k^\top\end{aligned}$$

- propagation of uncertainty to output using best estimate
- kalman gain
- update mean by projecting error on kalman gain
- update cov by propagating uncertainty on measurement backward

Kalman filter

$$\text{while no measurement: predict} \begin{cases} \hat{\boldsymbol{m}}_k = \mathbf{A}_k \boldsymbol{m}_{k-1} \\ \hat{\mathbf{P}}_k = \mathbf{A}_k \mathbf{P}_{k-1} \mathbf{A}_k^\top + \mathbf{Q}_{k-1} \end{cases}$$

$$\text{predict measurement} \begin{cases} \hat{\boldsymbol{y}}_k = \mathbf{H}_k \hat{\boldsymbol{m}}_k \\ \hat{\mathbf{S}}_k = \mathbf{H}_k \hat{\mathbf{P}}_k \mathbf{H}_k^\top + \mathbf{R}_k \end{cases}$$

$$\text{on measurement: update} \begin{cases} \mathbf{K}_k = \hat{\mathbf{P}}_k \mathbf{H}_k^\top \hat{\mathbf{S}}_k^{-1} \\ \boldsymbol{m}_k = \hat{\boldsymbol{m}}_k + \mathbf{K}_k \left(\boldsymbol{y}_k - \hat{\boldsymbol{y}}_k\right) \\ \mathbf{P}_k = \hat{\mathbf{P}}_k - \mathbf{K}_k \hat{\mathbf{S}}_k \mathbf{K}_k^\top \end{cases}$$

Implement these functions in a programming language:

- `m, P` $\longleftarrow$ `kf_predict(m, P, A, Q)`
- `y, S` $\longleftarrow$ `kf_measure(m, P, H, R)`
- `m, P` $\longleftarrow$ `kf_update(y, m, P, H, R)`

Note which matrices participate in each function. Update does not depend on $\mathbf{Q}$. How would you determine which states are most sensitive to the residuals $\boldsymbol{y}_k - \hat{\boldsymbol{y}}_k$?

Data driven models

What's a data driven model?

Data-driven as complementary to mechanistic. The model does not intentionally exploit prior knowledge about the **data generating process**.

If prior knowledge exists it desirable to consider it, but:

- Too much work
- Ignorance of methods

In science, efforts should be taken to exploit the prior knowledge.

There **are not** intrinsic data-driven models.

Linear regression can be mechanistic in some context, e.g. for an ideal gas at constant volume, pressure and temperature are linearly related:

$$PV = nRT$$

What prior knowledge do you have about:

- Falling objects
- The motion of planets in their orbits
- The behavior of bacteria

Mention a problem for which you would have loads of prior knowledge.

Linear regression

model : $y_k = t_k a + b + \epsilon_k$

Batch

$$\mathbf{D} = \begin{bmatrix} t_1 & 1 \\ & \vdots \\ t_N & 1 \end{bmatrix}, \quad \boldsymbol{y}^\top = \begin{bmatrix} y_1 & \cdots & y_N \end{bmatrix}$$

$$\mathbf{D} \begin{bmatrix} a \\ b \end{bmatrix} = \boldsymbol{y} + \boldsymbol{\epsilon}$$

Recursive (assumes ordered correlates)

$$\boldsymbol{x}_k = \begin{bmatrix} a \\ b \end{bmatrix}$$

$$\mathbf{A} = \mathbf{I}, \quad \mathbf{Q} = \mathbf{0}$$

$$\mathbf{H} = \begin{bmatrix} \Delta t & 1 \end{bmatrix}, \quad \mathbf{R} = \boldsymbol{\Sigma}_\epsilon$$

What would be the batch and recursive for of $y_k = t_k a_k + b + \epsilon_k$.
What would be the measurement model if correlates are not ordered?
How can you get rid of explicit t_k in the model? (t_k is in principle unbounded!)

Linear regression

model : $y_k = t_k a + b + \epsilon_k$

Batch

$$\mathbf{D} = \begin{bmatrix} t_1 & 1 \\ & \vdots \\ t_N & 1 \end{bmatrix}, \quad \boldsymbol{y}^\top = \begin{bmatrix} y_1 & \cdots & y_N \end{bmatrix}$$

$$\mathbf{D}\begin{bmatrix} a \\ b \end{bmatrix} = \boldsymbol{y} + \boldsymbol{\epsilon}$$

Recursive (assumes ordered correlates)

$$\boldsymbol{x}_k = \begin{bmatrix} a \\ b \end{bmatrix}$$

$$\mathbf{A} = \mathbf{I}, \quad \mathbf{Q} \neq \mathbf{0}$$

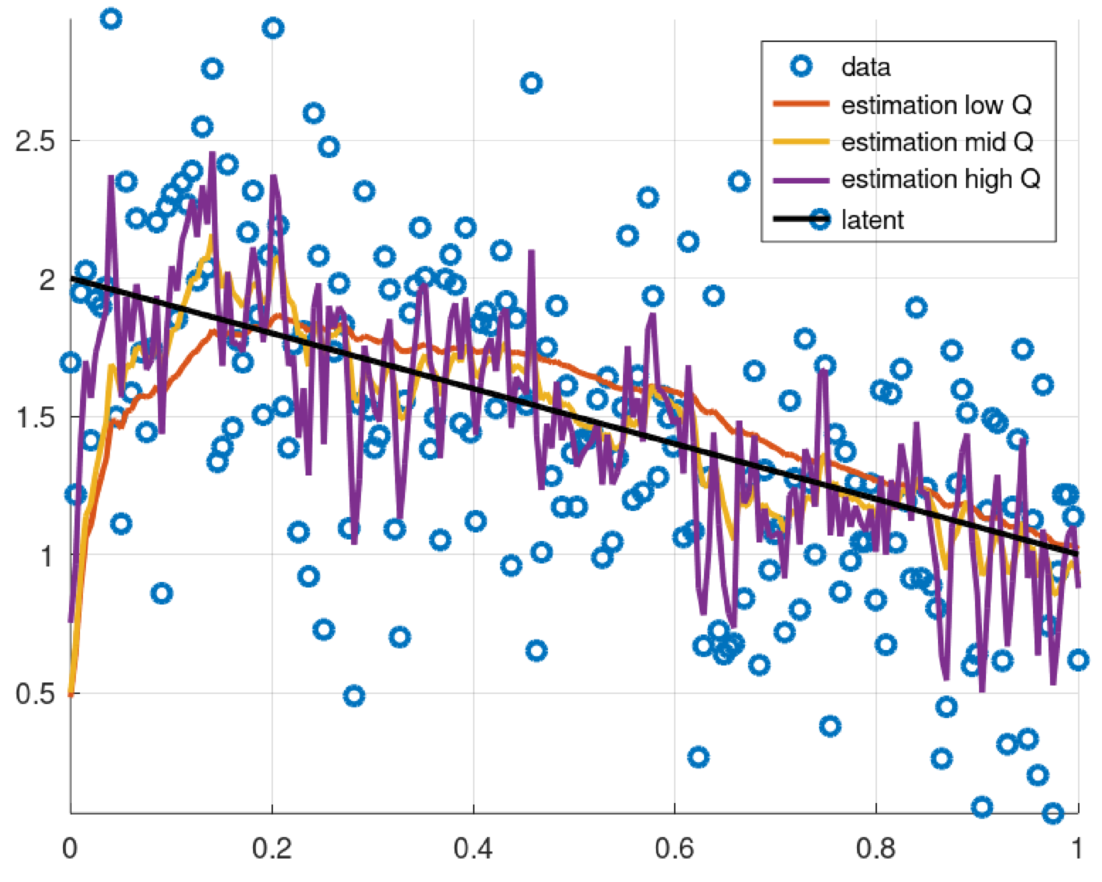
$$\mathbf{H} = \begin{bmatrix} \Delta t & 1 \end{bmatrix}, \quad \mathbf{R} = \boldsymbol{\Sigma}_\epsilon$$

See `s_filtering_interactive.py` (time varying linear regression)

The model without process noise does not update the initial values. By setting a symmetric non-negative process noise covariance the states update.

Linear regression

model : $y_k = t_k a + b + \epsilon_k$



source: `s_recursive_linreg.m`

Polynomial regression

model : $y_k = a_n t_k^n + \cdots + a_1 t_k + a_0 + \epsilon_k$

Batch

$$\mathbf{D} = \begin{bmatrix} t_1^n & \cdots & t_1 & 1 \\ & \vdots & & \\ t_N^2 & \cdots & t_N & 1 \end{bmatrix}, \quad \boldsymbol{y}^\top = \begin{bmatrix} y_1 & \cdots & y_N \end{bmatrix}$$

$$\mathbf{D} \begin{bmatrix} a \\ b \end{bmatrix} = \boldsymbol{y} + \boldsymbol{\epsilon}$$

Recursive

$$\boldsymbol{x}_k = \boldsymbol{a}$$
$$\mathbf{A} = \mathbf{I}, \quad \mathbf{Q} \neq \mathbf{0}$$
$$\mathbf{H}_k = \begin{bmatrix} t_k^n & \cdots & t_k & 1 \end{bmatrix}, \quad \mathbf{R} = \boldsymbol{\Sigma}_\epsilon$$

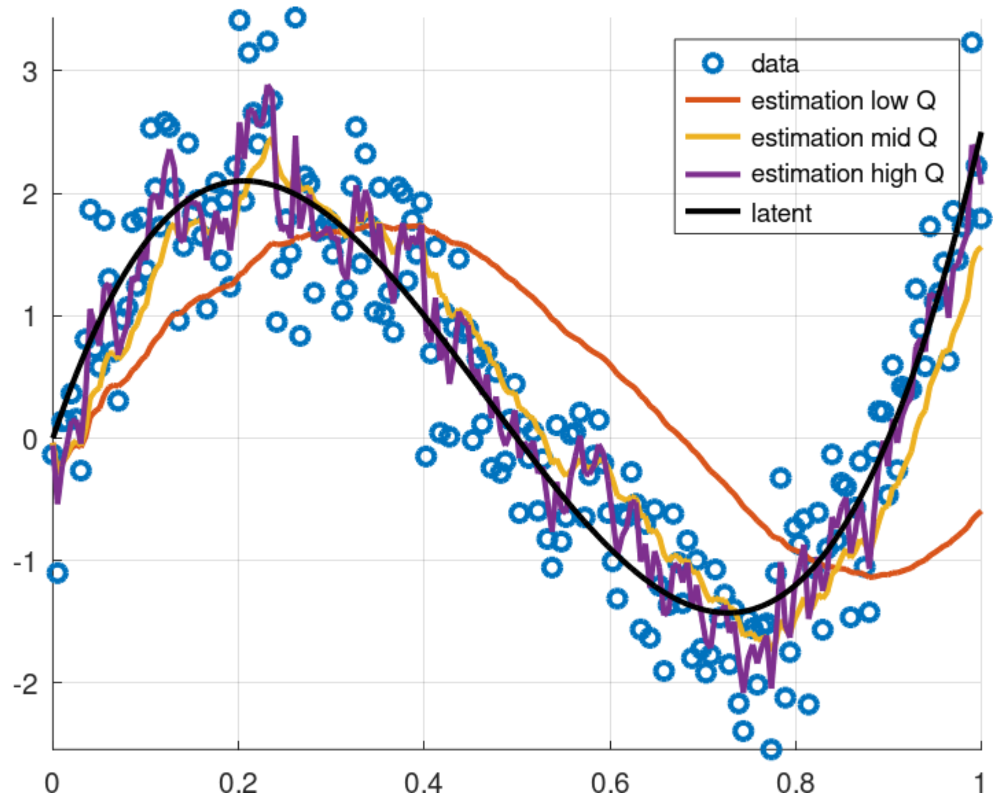
What is the issue with this measurement matrix? How would you avoid this issue?

$$\boldsymbol{x} = \begin{bmatrix} a_0 \\ \vdots \\ a_n \end{bmatrix}$$
$$\mathbf{A} = \mathbf{I} + \Delta t \mathbf{C}, \quad C_{ij} = \delta_{i(j+1)}$$

See `s_polyreg.m`

Polynomial regression

model : $y_k = a_n t_k^n + \cdots + a_1 t_k + a_0 + \epsilon_k$



source: *s_recursive_linreg.m*

Delayed regression

model : $y_k = w_1 y_{k-1} + \cdots + w_d y_{k-d} + \epsilon_k$ (auto-regressive model: AR)

Batch

$$\mathbf{D} = \begin{bmatrix} y_{d+1-1} & \cdots & y_1 \\ & \vdots & \\ y_{N-1} & \cdots & y_{N-d} \end{bmatrix}, \quad \boldsymbol{y}^\top = \begin{bmatrix} y_0 & \cdots & y_N \end{bmatrix}$$

$$\mathbf{D}\boldsymbol{w} = \boldsymbol{y}_{(d+1):N} + \boldsymbol{\epsilon}$$

Recursive

$$\begin{aligned} \boldsymbol{x}_k &= \boldsymbol{w}_k \\ \mathbf{A} &= \mathbf{I}, \quad \mathbf{Q} \neq \mathbf{0} \\ \mathbf{H}_k &= \begin{bmatrix} y_{k-1} & \cdots & y_{k-d} \end{bmatrix}, \quad \mathbf{R} = \boldsymbol{\Sigma}_\epsilon \end{aligned}$$

The rows of the design matrix are moving windows.

Extreme learning machines

Any function of the independent variable can be put in the design matrix:

$$\mathbf{D}_{k:} = \left[\phi_n(t_k) \;\cdots\; \phi_1(t_k)\right]$$

which corresponds to the measurement matrix, i.e.

$$\mathbf{H}_k = \mathbf{D}_{k:}$$

combined with the drift dynamic model

$$\boldsymbol{x}_k = \boldsymbol{w}_k$$
$$\mathbf{A} = \mathbf{I}, \quad \mathbf{Q} \neq \mathbf{0}$$

gives the recursive version.

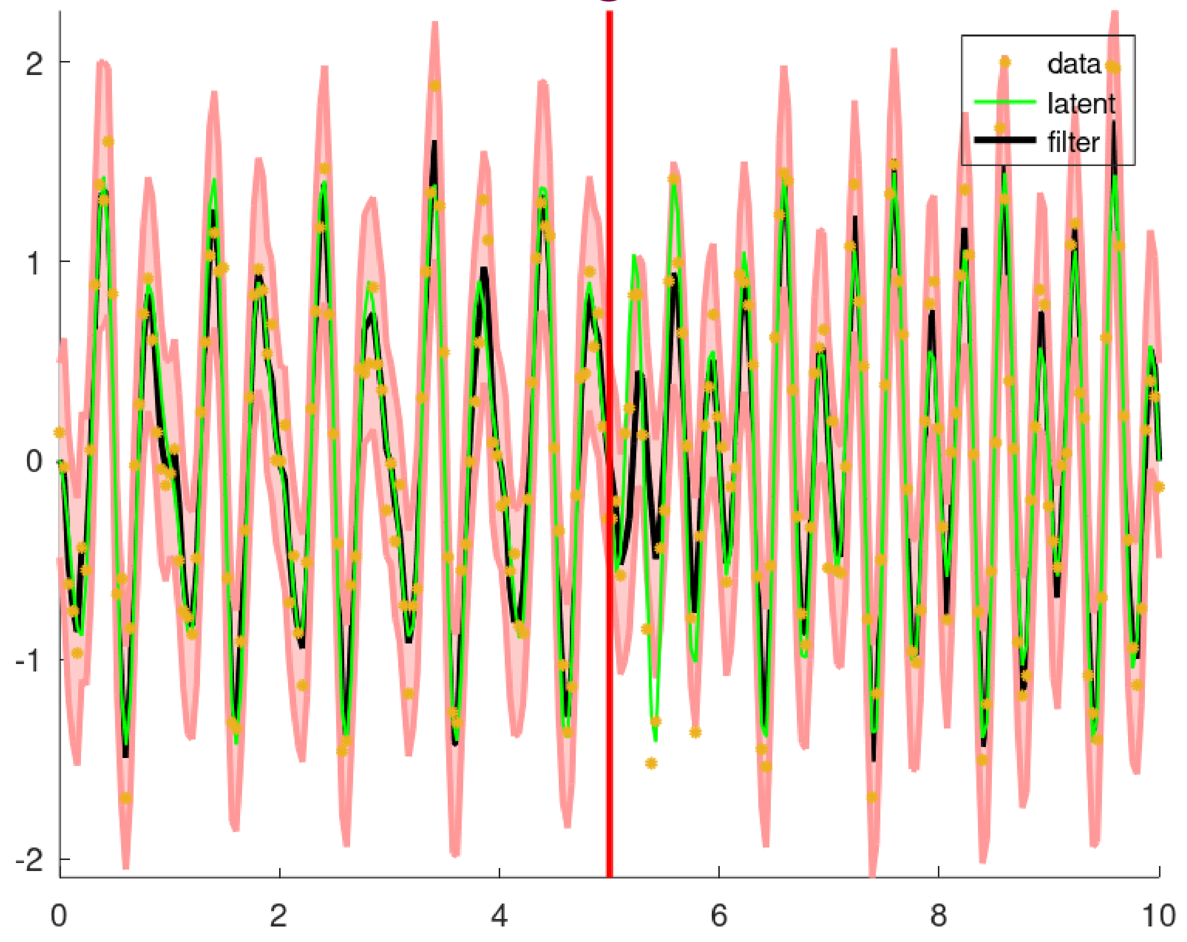
Properties of the function set $\{\phi_i\}$ can be used to write a more stable model, e.g. polynomials.

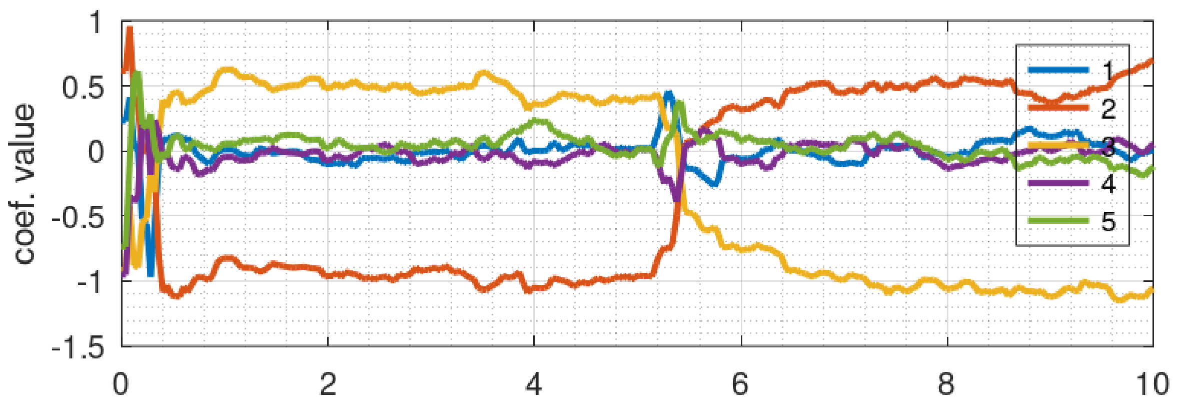
See `s_fourier.m` and `s_fourier_adaptive.m` for frequency tracking.

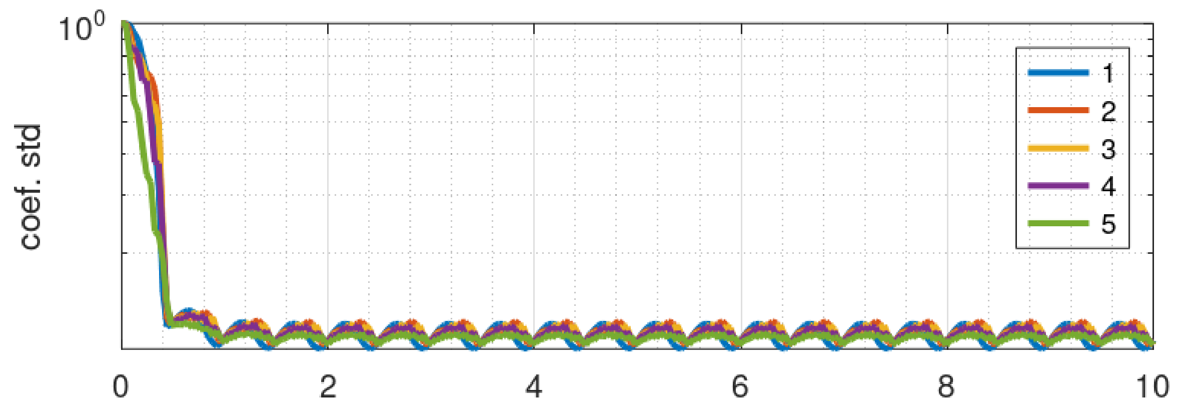
In machine learning $\mathbf{H}_k$ is also known as a feature vector.

The only requisite of KF is that the models (dynamic and measurement) are linear in the states.

Extreme learning machines







source: *s_fourier_adaptive.m*

The process noise allows the states to be adapted online. This means that the amplitude of each basis function is updated at each iteration. A quick change in the coefficients is picked up quickly (depending on process noise level).

Here the amplitudes of frequency 2 and 3 got exchanged.

Parameter estimation I

Augmented state

Augment your state with the parameters with constant noisy dynamics (drift model)

$$^{\text{extra}}x_k = {}^{\text{extra}}x_{k-1} + \epsilon_k, \quad \epsilon_k \sim \mathcal{N}(0, \sigma^2_{\text{extra}})$$

Works out-of-the-box if resulting model is linear in parameters, i.e. if we have in the dynamics a term of the form ax_{k-1}, adding a to the state will render the model non-linear.

σ_{extra} controls how "reactive" or "nervous" the parameter is. Larger values, quicker adaptation, larger confidence intervals.

See `s_fourier_adaptive.m`

Frequently there is a work-around to the "non-linearization" problem, imagination is the limit.

Batch regression of historical data

If you have historical data (not growing, not online) batch analyses could be performed.

Consider data $\left\{(t_i, \boldsymbol{x}_i)\right\}_{i=1}^{N}$

The model

$$\boldsymbol{x}_k = \mathbf{A}\boldsymbol{x}_{k-1}$$

can be learned from the data using linear regression.

For an ODE model

$$\dot{\boldsymbol{x}} = \mathbf{B}\boldsymbol{x}$$

Estimate the time derivative $\dot{\boldsymbol{x}}$ from the data, then do linear regression.

Nonlinear models can also be learned this way.

Once the dynamics is identified by a batch method, apply the KF formalism and continue recursively.

There is plenty of methods of learning dynamical systems from data, sometimes called "data-driven dynamical systems" or "system identification", and probably by other names depending on the community.
Some references:

- Functional Data Analysis (2005) by J. O. Ramsay and B. W. Silverman
- Gaussian Processes for Machine Learning (2006) by Carl Edward Rasmussen and Christopher K. I. Williams
- Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control (2019) by Steven L. Brunton and J. Nathan Kutz

Wrap-up

Kalman filter

$$\text{while no measurement: predict} \begin{cases} \hat{\boldsymbol{m}}_k = \mathbf{A}_k \boldsymbol{m}_{k-1} \\ \hat{\mathbf{P}}_k = \mathbf{A}_k \mathbf{P}_{k-1} \mathbf{A}_k^\top + \mathbf{Q}_{k-1} \end{cases}$$

$$\text{predict measurement} \begin{cases} \hat{\boldsymbol{y}}_k = \mathbf{H}_k \hat{\boldsymbol{m}}_k \\ \hat{\mathbf{S}}_k = \mathbf{H}_k \hat{\mathbf{P}}_k \mathbf{H}_k^\top + \mathbf{R}_k \end{cases}$$

$$\text{on measurement: update} \begin{cases} \mathbf{K}_k = \hat{\mathbf{P}}_k \mathbf{H}_k^\top \hat{\mathbf{S}}_k^{-1} \\ \boldsymbol{m}_k = \hat{\boldsymbol{m}}_k + \mathbf{K}_k \left(\boldsymbol{y}_k - \hat{\boldsymbol{y}}_k\right) \\ \mathbf{P}_k = \hat{\mathbf{P}}_k - \mathbf{K}_k \hat{\mathbf{S}}_k \mathbf{K}_k^\top \end{cases}$$

Implement these functions in a programming language:

- `m, P` $\longleftarrow$ `kf_predict(m, P, A, Q)`
- `y, S` $\longleftarrow$ `kf_measure(m, P, H, R)`
- `m, P` $\longleftarrow$ `kf_update(y, m, P, H, R)`

Note which matrices participate in each function. Update does not depend on $\mathbf{Q}$. How would you determine which states are most sensitive to the residuals $\boldsymbol{y}_k - \hat{\boldsymbol{y}}_k$?

Extreme learning machines

Any function of the independent variable can be put in the design matrix:

$$\mathbf{D}_{k:} = \left[\phi_n(t_k) \ \cdots \ \phi_1(t_k)\right]$$

which corresponds to the measurement matrix, i.e.

$$\mathbf{H}_k = \mathbf{D}_{k:}$$

combined with the drift dynamic model

$$\begin{gathered} \boldsymbol{x}_k = \boldsymbol{w}_k \\ \mathbf{A} = \mathbf{I}, \quad \mathbf{Q} \neq \mathbf{0} \end{gathered}$$

gives the recursive version.

Properties of the function set $\{\phi_i\}$ can be used to write a more stable model, e.g. polynomials.

See `s_fourier.m` and `s_fourier_adaptive.m` for frequency tracking.

In machine learning $\mathbf{H}_k$ is also known as a feature vector.

The only requisite of KF is that the models (dynamic and measurement) are linear in the states.